# Additive Counterfactuals for Explaining Link Predictions on Knowledge Graphs

Roberto Barile[1][0009−0007−3058−8692], Claudia d'Amato[1,2][0000−0002−3385−987X], and Nicola Fanizzi[1,2][0000−0001−5319−7933]

[1] Dipartimento di Informatica – University of Bari Aldo Moro, Italy
{roberto.barile, claudia.damato, nicola.fanizzi}@uniba.it
[2] CILA – University of Bari Aldo Moro, Italy

**Abstract.** The goal of *Link Prediction* is to predict missing facts in *Knowledge Graphs* that are inherently incomplete. *Embedding Models* are generally adopted for this purpose since they are effective and scalable. However, they lack both interpretability and, more importantly, explainability, which is crucial in many tasks and domains. To fill this gap, post-hoc explanations are often computed. A post-hoc explanation for an embedding-based link prediction typically consists of discovering facts that made the prediction possible. Methods that can yield such explanations tend to provide *subtractive counterfactual explanations*, i.e., identify facts whose removal has the greatest impact on the predictions. However, since KGs are incomplete, there may be facts that are missing in the KG but useful for the explanations. Therefore, we formalize a new complementary approach based on the generation of plausible and meaningful *additional facts* to be used for providing explanations. Specifically, we propose Imagine, that can generate *additive counterfactual explanations*, by identifying the *additional facts* that most affect predictions. It builds on a *post-training* technique, which is used to assess the impact of adversarial modifications of the knowledge graph, and on *Graph Summarization*, that is used for identifying plausible additions. We present a comparative experimental study that proves the effectiveness of the proposed solution through quantitative and qualitative evaluations.

## 1 Introduction

*Knowledge Graphs* (KGs) [18] have emerged as tools for representing, navigating, and querying the growing flood of data by making knowledge in diverse domains accessible to both humans and machines. A KG is a multi-relational graph composed of entities and relations, represented as nodes and edges, respectively. KGs are often integrated with ontologies that formally define classes and relations, allowing for advanced inference capabilities. There are several examples of large KGs, including enterprise products [33,14] and open collections [7,2,28].

Despite their proven utility, the inherent incompleteness of KGs often complicates their use, also as a consequence of the *Open World Assumption* (OWA)

typically made with KGs, as they result from complex, incremental, and distributed building processes [18]. This underpins the importance of *Link Prediction* (LP) and *Triple Classification* tasks, which aim at completing KGs by inferring missing facts and deciding on the truth of triples, respectively.

In this paper, we focus on LP that is generally tackled by means of *Machine Learning* (ML) methods grounded in *Knowledge Graph Embeddings* (KGEs) [30], which gained prominence because of their superior scalability. KGE models are based on vectors in low-dimensional spaces learned to represent elements in the KGs: they allow complex tasks to be framed as linear algebra operations. However, such models tend to operate as "opaque boxes" whose predictions are difficult to explain, potentially undermining their credibility and the confidence in their predictions. This opacity can be particularly problematic in fields where understanding the predictions is critical, such as finance, healthcare, and pharmacology. For example, relations among certain drugs and side effects may be predicted, hence it becomes crucial having clues in order to possibly identify potential associations [27] and/or understanding if such predictions may influence decisions on investment in a specific drug research.

In this regard, *Explainable Artificial Intelligence* (XAI) [17] methods come into focus to enhance the transparency and comprehensibility of ML models. They can be divided into 1) *post-hoc* methods, useful for computing explanations given the predictions, and 2) *clear box* methods, that can produce predictions along with their explanations [17]. We will focus on the former approach because, unlike the latter, these methods can be independent of the LP method adopted. In the *post-hoc* setting, one of the most valuable types of explanation is the *Counterfactual Explanation* (CE), which indicates potential changes in the knowledge source that could alter the outcome [16]. CEs can be divided into *Subtractive CEs* (SCEs), for identifying salient removal changes in the knowledge source, and *Additive CEs* (ACEs), for identifying significant additional information [9]. A *post-hoc* explanation for a prediction typically consists of a specific set of facts that allow such an inference.

Several promising *post-hoc* explanation methods have emerged, e.g., [36], Kelpie [31] and Kelpie++ [4], but all of them are grounded on SCEs identifying facts that most significantly affect the predictions if removed. Hence, only existing facts in the KG are considered for generating explanations for the computed predictions. However, as KGs are to be considered inherently incomplete, there may be facts that are missing from the KG that are actually useful for building the explanations. In this perspective, ACEs could be computed for identifying *additional facts* that would most affect the predictions if inserted, thus providing complementary explanations to SCEs. This view is also coherent with the OWA characterizing the KGs in logic contexts, for which there could be plausible facts missing in the KG. Even more so, cognitive studies proved that, when considering explanations, people tend to prefer ACEs to SCEs because the former foster creative thinking [9].

Motivated by these observations, we formalize a new *post-hoc* explanation method that is grounded in the generation of plausible and meaningful *addi-*

*tional facts* to be employed for providing explanations. Specifically, we propose IMAGINE a framework for providing *Additive Counterfactual Explanations* for LP, which is able to generate them, by identifying the *additional facts* that most affect the predictions. It builds on a *post-training* technique introduced in [31] and that is used for assessing the impact of adversarial modifications of the KG, and on *Graph Summarization* [10], that is adopted for identifying plausible *additional facts*.

It is worth noting that the explanations computed by IMAGINE can be complementary to SCEs, since both SCEs and ACEs can help to understand the rationale behind the predictions by analyzing how the predictions are affected, that is, either improve or worsen. Ideally, ACEs are those showing to improve the predictions. For this purpose, we perform a comparative experimental analysis in order to verify this hypothesis. It is also worthwhile to be mentioned that valuable ACEs that are computed by IMAGINE could be exploited for understanding how the KG could be refined by optimizing the LP performances. Moreover, even if the generated *additional facts* may not be valid, they still provide valuable insights, as they help to understand how the model behaves under different conditions.

The contributions of the paper are the following:

- we formalize IMAGINE, to the best of our knowledge, the first solution for computing *Additive Counterfactual Explanations* to LP in KGs, to be meant as complementary to solutions for generating SCEs
- IMAGINE offers insights for understanding how the KG could be refined by optimizing the performance of LP
- we experimentally prove, via quantitative and qualitative evaluation, that IMAGINE provides valuable explanations, by adopting state-of-the-art evaluation protocols
- we experimentally prove that ACEs and SCEs approaches are complementary and argue on the need for the formalization of a new unified framework.

The rest of the paper is organized as follows. § 2 reviews the most recent and effective approaches for explaining LPs. § 3 presents fundamentals for the paper understanding. § 4 details our new solution, IMAGINE, that computes explanations to LP results by adopting ACEs. § 5 illustrates the comparative experimental study, proving the effectiveness of IMAGINE via quantitative and qualitative evaluation. § 6 summarizes the achievements and challenges and outlines future research directions.

## 2 Related Works

In this section, we survey *post-hoc* methods for explaining LP on KGs. The firstly emerged methods[36,29] explain a prediction by providing a single fact, that is a statement: subject, predicate, object. Specifically, in [36] the fact whose insertion or removal most significantly poisons the prediction is returned, while CRIAGE [29], employs approximated *Influence Functions* [23], and focuses on

facts featuring as object either the subject or the object of the prediction. Hence, we regard these methods as providing SCEs.

With the goal of targeting more complex explanations, methods explaining a prediction by providing a path from the subject to the object of the prediction have been proposed. CrossE [38] and ApproxSemanticCrossE [11] find paths exploiting the KG topology and (semantic) similarity of entities and/or relationships with respect to the predicted fact, while, LinkLogic [22] adopts a perturbation based approach.

More recent methods explain a prediction by returning a specific set of relevant facts, rather than a single fact or a path. Kelpie [31] employs a novel *post-training* process for adversarial re-training of the KGE adopted for the prediction, but that can be tailored to different KGE models. Kelpie++ [4] integrates *Graph Summarization* into Kelpie making it more efficient, while enhancing the effectiveness of the explanations and allowing the presentation/inspection of the explanations at different level of granularity. Both Kelpie and Kelpie++ provide two synergistic types of explanations: *necessary* and *sufficient* explanations. A *necessary* explanation for a prediction is a set of facts without which the model would not have made the same inference. Such *necessary* explanations fall into the SCEs category. In contrast, a *sufficient* explanation is a set of facts that makes the model able to replicate the given prediction for other entities. The process for finding *sufficient* explanations involves generating *additional facts* by starting with those involving the subject of the prediction and then corrupting these facts with other entities. Similarly, KGEx [3] utilizes subgraph sampling and *Knowledge Distillation*, while [5] provides explanation by performing *abduction* on a logical theory learned through an ML approach.

Differently, KE-X [39] frames KGE score functions in a *Message Passing* framework to identify the facts that maximize the *Information Gain* with respect to the prediction. Notably, GEnI [1] also includes schema axioms in the explanations by evaluating whether predicate embeddings satisfy certain numerical properties. However, it is restricted to translational and bilinear KGE models and as such it is not fully model agnostic. KGExplainer [26] uses a greedy search algorithm based on perturbations. All the aforementioned methods build explanations from facts in the KG, whereas our goal is to employ missing but reasonable facts. In *sufficient* explanations, facts missing in the KG are involved. Yet the original facts from which *additional facts* are derived as perturbations ultimately contribute to the explanations, but not the *additional facts*. Instead, we aim to provide explanations consisting of actual *additional facts*.

Further methods providing explanations not consisting in a specific set of relevant facts or a path have been also proposed. In [21] *Horn Clauses* are mined to explain predictions, while FeaBI [19] can build interpretable vectors from entity embeddings based on propositional features extracted from the KG via *Feature Selection*. In this category, also [6], XTransE [37], and GNNExplainer [35] tailored for *Graph Neural Networks* are worthwhile to be mentioned, but they adopt a *clear box* approach rather than the *post-hoc* approach that is our target.

Finally, other *post-hoc* methods that are independent of the task requiring the explanation have been proposed. They can be categorized along their yielding: a) *saliency explanations*, to identify relevant input features as explanations; or b) *prototypes*, to identify training samples as explanations. Methods grounded on *saliency explanation*, such as SHAP [25], are popular, but struggle with predictions based on KGEs whose numerical features lack interpretability. Solutions based on *prototypes* are more suitable for LPs, but often *Influence Functions* are employed to the purpose [20] which pose scalability issues as in CRIAGE.

In contrast with the surveyed methods, IMAGINE focuses on CEs rather than *clear-box* approaches and *post-hoc* methods alternative to CEs, and as such it is applicable to any KGE model. Moreover, being focused on ACEs, IMAGINE is able to exploit the inherent incompleteness of KGs for explanations purposes.

## 3 Fundamentals

In this section, we present the fundamentals that are functional for the full understanding of proposed explanation solution IMAGINE. In § 3.1 we introduce KGs more formally and the basics of KGE methods. In § 3.2 we recall the notion of quotient graph that is employed by IMAGINE for performing graph summarization, which is exploited for identifying plausible additions.

### 3.1 Knowledge Graphs and Embedding-based Link Prediction

A KG is a graph-based data structure $\mathcal{G}(\mathcal{V}, \mathcal{R})$ with $\mathcal{V}$ representing a set of nodes, also known as entities, and $\mathcal{R}$ representing a set of predicates. In the adopted RDF model, a KG is a collection of triples in the format $\langle s, p, o \rangle$, i.e., *subject*, *predicate*, and *object* where $s, o \in \mathcal{V}$ and $p \in \mathcal{R}$.

Various models have been proposed for representing KGs in low-dimensional vector spaces, by learning for each entity and predicate in the KG a unique numerical vector (or *embedding*) in a given space. Several embedding spaces can be employed, e.g., real, point-wise, complex, discrete, Gaussian, manifold. Without loss of generality, in the following we focus on real embeddings (vectors will be denoted in **bold**).

All these models represent each entity $e \in \mathcal{V}$, and each predicate $p \in \mathcal{R}$ by means of an embedding vector $\mathbf{e} \in \mathbb{R}^k$, and $\mathbf{p} \in \mathbb{R}^i$, respectively, where $k, i \in \mathbb{N}$ are hyperparameters. Moreover, each model is associated to a *scoring function* $f : \mathbb{R}^k \times \mathbb{R}^i \times \mathbb{R}^k \to \mathbb{R}$: for each triple $\langle s, p, o \rangle$, the score $f(\langle s, p, o \rangle)$ measures the likelihood of such statement. In the following, we report formulations where higher values convey more plausibility; symmetric formulations can be derived for models where lower scores convey higher likelihood. The embeddings and parameters are learned from the KG by minimizing a loss function grounded on $f$. To this purpose, the set of triples encoded by $\mathcal{G}$ is divided into a training set $\mathcal{G}_{train}$, a validation set $\mathcal{G}_{val}$ and a test set $\mathcal{G}_{test}$. Beyond entity and predicate embeddings, models can also learn shared parameters that are not explicitly connected to any KG element, and similar to the weights of neural layers.

Given an incomplete triple $\langle s, p, ?\rangle$, LP is performed by computing

$$o = \arg\max_{e \in \mathcal{V}} f(\langle s, p, e\rangle)$$

Moreover, the *rank* of a triple $\langle s, p, o\rangle$ in $\mathcal{G}_{test}$ can be computed as:

$$\text{rank}(\langle s, p, o\rangle) = |\{e \in \mathcal{V} \mid f(\langle s, p, e\rangle) >= f(\langle s, p, o\rangle)\}|$$

Computing the rank of each triple in $\mathcal{G}_{test}$ is required for evaluating the LP performance.

### 3.2 Quotient Graphs and Simulations

In the context of KGs, quotient graphs aim at summarizing the data graph with a higher-level topology [10]. These are based on the notion of quotient set. Given a set $X$ and an equivalence relation $\sim$, $X$ is partitioned into disjoint subsets of equivalent elements. The *quotient set* $X/_\sim$ contains all such subsets, the *equivalence classes*. Then, considering the KG $\mathcal{G}(\mathcal{V}, \mathcal{R})$, its *quotient graph* $\mathcal{Q}(\mathcal{V}/_\sim, \mathcal{R})$ w.r.t. a relation $\sim$ over $\mathcal{V}$ will have $\mathcal{V}/_\sim$, quotient set of $\mathcal{V}$ w.r.t. $\sim$, as its node set and the same predicate set $\mathcal{R}$. Note that the triples in $\mathcal{Q}$ can be defined in different ways, depending on the desired level of preservation of the structure in $\mathcal{G}$.

In our approach, we will focus on a simulation of the original $\mathcal{G}$, requiring a proper level of abstraction, with no need to preserve its complete structure. A *simulation* [18] is a binary relation from a graph $\mathcal{G}$ to a graph $\mathcal{G}'$ that ensures that any path between connected nodes in $\mathcal{G}$ also exists in $\mathcal{G}'$.

Formally: a *simulation* from a graph $\mathcal{G}(\mathcal{V}, \mathcal{R})$ to a graph $\mathcal{G}'(\mathcal{V}', \mathcal{R}')$ is a relation $S \subseteq \mathcal{V} \times \mathcal{V}'$ such that for each triple $\langle x, y, z\rangle$ in $\mathcal{G}$, if $x\ S\ x'$ exists for some $x' \in \mathcal{V}'$, then there exists $z' \in \mathcal{V}'$ such that $z\ S\ z'$ and $\langle x', y, z'\rangle$ is a triple in $\mathcal{G}'$. $\mathcal{G}'$ *simulates* $\mathcal{G}$ when a simulation exists from $\mathcal{G}$ to $\mathcal{G}'$. Adopting simulation, a triple $\langle X, y, Z\rangle$ exists in $\mathcal{Q}$ if and only if, for some $x \in X$ and $z \in Z$, the triple $\langle x, y, z\rangle$ is in the original graph $\mathcal{G}$.

An alternative way for defining the triples in $\mathcal{Q}$ is bisimulation [4]. It is a stricter condition than simulation, indeed, adopting bisimulation, a triple $\langle X, y, Z\rangle$ exists in $\mathcal{Q}$ if and only if, for each $x \in X$ and $z \in Z$, the triple $\langle x, y, z\rangle$ is in the original graph $\mathcal{G}$.

## 4 The Proposed Approach

We introduce IMAGINE, a search-based method for generating the ACEs for link predictions on KGs. ACEs consist of *additional triples*, i.e., those that are neither explicitly stated in, nor entailed by, the KG, yet are not assumed to be false under OWA.

Specifically, given a true triple $t = \langle s, p, o\rangle$, IMAGINE returns as explanation a set of *additional triples* featuring $s$ that is relevant for $t$, i.e., leading the LP

method to modify its rank. Moreover, we consider an *additional triple* as useful if it is likely part of an explanation.

IMAGINE builds on KELPIE [31] and KELPIE++ [4], that generate effective CEs involving multiple triples, apply to arbitrary KGE models and offer resources for the reproducibility of experiments. As for them, IMAGINE is a *post-hoc* explanation method applicable to any KGE model, but differently from KELPIE and KELPIE++, that are grounded on computing SCEs, to the best of our knowledge, IMAGINE is the first solution providing ACEs.

IMAGINE is organized around the following four main components:

- Triple Builder that generates a set of useful *additional triples* featuring $s$;
- Pre-Filter that selects the most useful *additional triples*;
- Explanation Builder that combines the pre-filtered triples into *candidate explanations* and identifies sufficiently relevant ones;
- Relevance Engine that computes the relevance of a *candidate explanation*.

IMAGINE overall architecture is similar to the one of the aforementioned systems, but a novel solution for the Triple Builder component is formalized, since it is the component in charge of generating *additional triples* by computing ACEs. Moreover, a modified version of the Relevance Engine is proposed by formalizing a new alternative way for computing the relevance score of candidate explanations. The Pre-Filter and Explanation Builder are adaptations of their counterparts in KELPIE and KELPIE++. Each of these components is described below.

### 4.1 Triple Builder

For its purpose, the Triple Builder relies on *Graph Summarization*, leveraging the notion of quotient graph (see § 3.2). Essentially, this component abstracts the KG and reconstructs the original one. The reconstruction contains *additional triples*, as the abstraction does not perfectly preserve the original structure. Specifically, each entity inherits triples from the other entities in its equivalence class. The rationale behind this component is that *additional triples* formed with entities in the same equivalence class of the subject of the prediction are more useful than those originating from entities of different types.

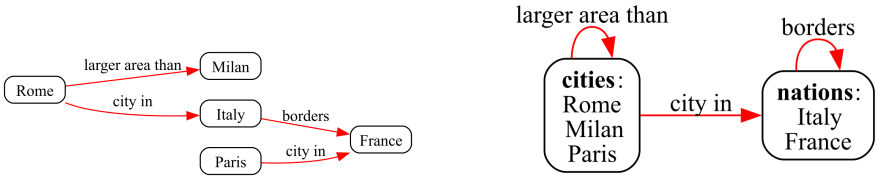
In Alg. 1 we formalize the logic of the Triple Builder. It requires as input the training KG, its quotient and the prediction to be explained. Moreover, we define $\mathcal{G}^s_{train}$ as the subgraph of $\mathcal{G}_{train}$ containing the triples featuring $s$ as subject or as object. The quotient graph $\mathcal{Q}_{train}$ of $\mathcal{G}_{train}$ is computed initially and independently of the specific prediction. IMAGINE computes it by exploiting a function $\tilde{Cl}$ to determine the equivalence relation. Such function $Cl$ is defined in [11] as returning the classes to which an entity can be proven to belong to; while, its approximated version $\tilde{Cl}$ simplifies the required reasoning service (*realization*).

As an example, we report in Fig. 1a a very simple KG regarding cities/nations and its quotient graph in Fig. 1b. Moreover, we consider the prediction $\langle Milan, located\ in, Italy \rangle$. Initially, the algorithm identifies the quotient node $S$

**Algorithm 1:** Generate additional triples featuring $s$

**Input:** triple $\langle s, p, o \rangle$; training graph $\mathcal{G}_{train}$; quotient graph $\mathcal{Q}_{train}$
**Output:** set of additional triples $\mathcal{A}^s$

1 $S \leftarrow$ get_quotient_node$(s, \mathcal{Q}_{train})$;
2 $triples \leftarrow \{\}$;
3 **foreach** $\langle S, r, O \rangle \in Q_s^{train}$ **do**
4  **if** $s \in S$ **then**
5  $\mathcal{A}^s \leftarrow \mathcal{A}^s \cup \{\langle s, r, o \rangle \mid o \in O\}$;
6  **if** $s \in O$ **then**
7  $\mathcal{A}^s \leftarrow \mathcal{A}^s \cup \{\langle o, r, s \rangle \mid o \in S\}$;
8 $\mathcal{A} \leftarrow \mathcal{A}^s \setminus \{\langle s, p, o \rangle\}$;
9 $\mathcal{A} \leftarrow \mathcal{A}^s \setminus \mathcal{G}^s_{train}$;




(a) (b)

Fig. 1: A very simple KG regarding cities/nations (a) and its quotient graph (b)

in $\mathcal{Q}_{train}$ containing the prediction's subject $s$ (Line 1). In our example, $s$ is *Milan*. Then, the output set $\mathcal{A}^s$ is initialized as an empty set (Line 2). The process then continues by analyzing each quotient triple $\langle S, p, O \rangle$ in $\mathcal{Q}^s_{train}$ (Line 3). Specifically, if $s$ is in $S$, then Triple Builder extends $\mathcal{A}^s$ with the triples $\langle s, r, o \rangle$ for all $o$ in $O$ (Lines 4-5). Conversely, if $s$ is in $O$, the triples $\langle o, r, s \rangle$ for all $o$ in $O$ are added to $\mathcal{A}^s$ (Lines 6-7). For instance, for the quotient triple $\langle \{Milan, Rome, Paris\}, city\ in, \{Italy, France\} \rangle$, the additional triples $\{\langle Milan, city\ in, Italy \rangle, \langle Milan, city\ in, France \rangle\}$ arise. Finally, the algorithm removes any redundant triples from $\mathcal{A}^s$. Specifically, it removes the input triple $t$ itself (Line 8) and, additionally, it removes the triples already occurring in $\mathcal{G}^s_{train}$ (Line 9).

### 4.2 Pre-Filter and Explanation Builder

The Pre-Filter aims at reducing the complexity for the following stages. Therefore, this component filters $\mathcal{A}^s$ to obtain $\mathcal{F}^s$ by selecting $k$ triples based on a topological measure of utility. Specifically, the utility of a triple $\langle s, p, e \rangle$ is the length of the shortest-path from $e$ to the $o$. The Explanation Builder, on the other hand, searches for the set of triples in $\mathcal{F}^s$ that make up an acceptable explanation $X^*$ in terms of a given relevance threshold. Specifically, it enumerates all possible candidate explanations (each denoted as $X$) as sets/combinations of the

**Algorithm 2:** Assess the *relevance* of a *candidate explanation*, I-mode

**Input:** prediction $\langle s, p, o \rangle$, candidate explanation $X$;
training graph $\mathcal{G}_{train}(\mathcal{V}, \mathcal{R})$, KGE model $m$;
**Output:** relevance $rel$

1 $\mathcal{V} \leftarrow \mathcal{V} \cup \{new\}$;
2 $\mathcal{G}^{new}_{train} \leftarrow$ get_triples($\mathcal{G}^{s}_{train}, new$);
3 **new** $\leftarrow$ rand_init();
4 **new** $\leftarrow$ post_train($\mathcal{G}^{new}_{train} \cup X$);
5 $rank_{new} \leftarrow m$.rank($\langle new, p, o \rangle$);
6 **new** $\leftarrow$ rand_init();
7 **new** $\leftarrow$ post_train($\mathcal{G}^{new}_{train}$);
8 $rank'_{new} \leftarrow m$.rank($\langle new, p, o \rangle$);
9 $rel \leftarrow (rank'_{new} - rank_{new})/(rank'_{new} - 1)$;

triples $\mathcal{F}^s$, then it determines whether any $X$ can be accepted as $X^*$ after invoking the Relevance Engine. Moreover, it features heuristic conditions to prune the space of candidate explanations.

In addition, IMAGINE features an extended version of the Explanation Builder, as suggested in KELPIE++. It includes a step that summarizes $\mathcal{F}^s$ to speed up the search and possibly lead to more effective explanations. The summarization can be performed using two different modes: *simulation* and *bisimulation*.

### 4.3 Relevance Engine

This component assesses the relevance of a candidate explanation $X$ by estimating the rank variation of the triple $\langle s, p, o \rangle$ after adding the triples in $X$ to $G_{train}$ and re-training the KGE model. Ideally, the relevance of $X$ should be assessed by re-training all the KGEs from scratch on the entire KG. However, this approach is impractical. Hence, we employ *post-training*, which is a scalable approximation of re-training introduced in KELPIE. It consists in adding a new entity to $G_{train}$ and learning the corresponding embedding using a set of samples obtained by adapting the triples in $G^s_{train}$ to the new entity, while keeping all the other embeddings and parameters fixed. The process of *post-training* a new entity is more efficient than the ideal complete re-training, as it solely optimizes one embedding on a limited set of triples.

We formalize the process of IMAGINE in Alg. 2. Firstly, the Relevance Engine adds a new entity called *new* to the training graph (Line 1). Note that, *new* is an actual entity that the algorithm adds to the set $\mathcal{V}$, rather than a variable representing an element of $\mathcal{V}$. Then, it computes the set $\mathcal{G}^{new}_{train}$ of triples for this entity, starting from $\mathcal{G}^s_{train}$ (Line 2). Specifically, for each triple $\langle s, r, q \rangle$ ($\langle q, r, s \rangle$) in $\mathcal{G}^s_{train}$, it adds to $\mathcal{G}^{new}_{train}$ the triple $\langle new, r, q \rangle$ ($\langle q, r, new \rangle$). Next, it initializes a random embedding formalized as **new** for such entity (Line 3). Then it optimizes this embedding by performing *post-training* on the set $\mathcal{G}^{new}_{train} \cup X$ (Line 4). Then, it computes the $rank_{new}$ of the triple $\langle new, p, o \rangle$, which is obtained by injecting the new entity in the prediction to explain (Line 5). Next, the Relevance Engine

**Algorithm 3:** Assess the *relevance* of a *candidate explanation*, W-mode

**Input:** prediction $\langle s, p, o \rangle$, candidate explanation $X$;
training graph $\mathcal{G}_{train}(\mathcal{V}, \mathcal{R})$, KGE model $m$;
**Output:** relevance $rel$

1 $\mathcal{V} \leftarrow \mathcal{V} \cup \{new\}$;
2 $\mathcal{G}^{new}_{train} \leftarrow$ get_triples($\mathcal{G}^{s}_{train}, new$);
3 **new** $\leftarrow$ rand_init();
4 **new** $\leftarrow$ post_train($\mathcal{G}^{new}_{train} \cup X$);
5 $rank_{new} \leftarrow m$.rank($\langle new, p, o \rangle$);
6 **new** $\leftarrow$ rand_init();
7 **new** $\leftarrow$ post_train($\mathcal{G}^{new}_{train}$);
8 $rank'_{new} \leftarrow m$.rank($\langle new, p, o \rangle$);
9 $rel \leftarrow rank_{new} - rank'_{new}$

can compute relevance by comparing $rank_X$ with the rank of the prediction. However, an additional step is made (as introduced in KELPIE) for making the process more robust to random fluctuations. Specifically, it re-initializes randomly the embedding **new** (Line 6). Then, it optimizes such embedding, but performing *post-training* on the set $\mathcal{G}^{new}$ rather than $\mathcal{G}^{new}_{train} \cup X$ (Line 7). Next, it computes again $rank_{new}$ (Line 8). Finally, it computes the relevance $rel$ as the rank improvement divided by the ideal rank improvement, which is $rank'_{new} - 1$ (Line 9). To clarify, for a triple with rank $k$, achieving a rank improvement of $k - 1$ leads to rank it as first, which is the ideal rank.

We also introduce a variant of IMAGINE that we dub W-IMAGINE (to signify *worsen*).

W-IMAGINE differs from IMAGINE solely in the Relevance Engine component. We can formalize the process related to W-IMAGINE in Alg. 3. The Relevance Engine involves a *post-training* process that is analogous to Alg. 2, but it computes relevance as a measure of rank worsening, specifically: $rank_{new} - rank'_{new}$. W-IMAGINE allows for assessing the impact of *additional triples* on top ranked predictions, while IMAGINE is tailored to those not ranked as first.

## 5 Experimental Evaluation

We illustrate the experimental evaluation that was carried out, specifying the experimental setup and presenting quantitative and qualitative results.

### 5.1 Experimental Setting

We performed the experiments on three datasets: DB50K [32], DB100K [13] and YAGO4-0 [4]. We report statistics on the datasets in Tab. 1. The common aspect of these datasets is that, along with the RDF triples, they contain OWL (specifically OWL2-DL) statements, including class assertions and other schema axioms involving classes and relationships. The integration of these datasets

Table 1: Statistics of the datasets

| | Entities | Relations | Train triples | Valid triples | Test triples |
|---|---|---|---|---|---|
| **DB50K** | 24620 | 351 | 32194 | 123 | 2095 |
| **DB100K** | 98776 | 464 | 587688 | 49172 | 49114 |
| **YAGO4-20** | 96910 | 70 | 555182 | 69398 | 69398 |

Table 2: Performance of the LP models

| | DB100K | | DB50K | | YAGO4-20 | |
|---|---|---|---|---|---|---|
| | *H@1* | *MRR* | *H@1* | *MRR* | *H@1* | *MRR* |
| ComplEx | 0.346 | 0.408 | 0.410 | 0.457 | 0.197 | 0.245 |
| ConvE | 0.301 | 0.363 | 0.112 | 0.144 | 0.151 | 0.197 |
| TransE | 0.061 | 0.143 | 0.282 | 0.367 | 0.048 | 0.095 |

with the corresponding ontologies is described in [4]. We employed the reasoner HermiT [15] offline to materialize the implicit class assertions in the KGs. Next, the function $\tilde{Cl}$ was implemented as a simple lookup over explicit class assertions.

Imagine supports any KGE model; therefore, we performed our experiments on TransE [8], ConvE [12] and ComplEx [34] as these represent the three major families of models: *Geometric*, *Deep Learning*, and *Tensor Decomposition* KGE models. Tab. 2 reports the LP performance of each model that we measured on each dataset in terms of typical measures, namely *Mean Reciprocal Rank* (*MRR*) and *Hits-at-1* (*H@1*) in their filtered variant.

We adopt a methodology building on the one introduced for the evaluation of Criage (also utilized for Kelpie and Kelpie++). It measures effectiveness as the impact on the LP performance achieved through the CEs. Both SCEs and ACEs can either improve or worsen such results, so we conduct experiments that measure effectiveness in terms of both improvement and worsening of LP performance. For evaluating the improvement, for each KGE model, we randomly sample a set $P$ of 100 test triples that are not ranked first by the LP method. After extracting explanations for all predictions in $P$, we add their triples and re-train the model. As the original model did not yield those predictions, their initial $H@1$ is 0.0, while their *MRR* is less than 1.0. However, if the extracted explanations are indeed able to increase the rank, the re-trained model should lead to improve both metrics. Therefore, the effectiveness is assessed as the increase in the $H@1$ and *MRR* over $P$. Conversely, for evaluating the worsening of LP results we select a set $P$ of 100 test set triples that are ranked first by the LP method, randomly, for each model. After extracting explanations for all predictions in $P$, we add their triples and re-train the KGE model. As the original model correctly led to those predictions, their initial values for both $H@1$ and *MRR* are 1.0. However, if the extracted explanations are indeed able to decrease the rank, the re-trained model should lead to a worsening of both metrics. Therefore, the effectiveness is assessed as the decrease in $H@1$ and *MRR*

over $P$. In both scenarios, the variations of the metrics are denoted as $\Delta H@1$ and $\Delta MRR$, respectively.

Ideally, ACEs are those leading to an improvement. To demonstrate that IMAGINE can generate valuable ACEs, we compare it to baseline methods that, similarly to IMAGINE, generates CEs aimed at improving LP results, but is based on existing triples and thus returns SCEs. This baseline methods, which we call I-KELPIE and I-KELPIE++ (to emphasize its focus on *improving* LP performance), are built on KELPIE and KELPIE++. Whilst KELPIE and KELPIE++ identify the sets of triples whose removal most decreases the ranks of the triples in $P$, I-KELPIE and I-KELPIE++ extract the sets of triples whose removal most increases the ranks of the triples in $P$. Intuitively, it identifies noisy triples, or those that encode information that is meaningful but unrelated to the predictions. The goal is to show that the *additional triples* generated by IMAGINE are indeed useful, as they contribute more significantly to improving LP performance than noise removal. We include experiments for each summarization approach adopted in the Explanation Builder: no summarization, *bisimulation*, and *simulation*. Essentially, we compare IMAGINE, I-KELPIE, and I-KELPIE++.

Even more so, for demonstrating that the ACEs generated by IMAGINE are valuable, we set up experiments measuring the worsening of LP performance. We compare W-IMAGINE with KELPIE and KELPIE++. We aim at proving that SCEs worsen LPs more than ACEs, thus implying that *additional triples* are not a source of noise to the extent that adding triples worsen LPs more than removing existing triples, even if in the Relevance Engine we focus on the rank decrease. We experiment with the different summarization solutions in the Explanation Builder. Essentially, we compare KELPIE, KELPIE++, W-IMAGINE, and W-IMAGINE++.

The code, the datasets, the trained models utilized in our study are openly accessible on GitHub[3] along with a detailed account of the hyperparameters adopted in all the phases of the experiments and a description of the execution environment.

### 5.2 Outcomes of the Evaluation

In Tab. 3 we report the effectiveness metrics for the experiments measuring the improvement of LP performance. IMAGINE consistently outperforms I-KELPIE and I-KELPIE++ in any setup of the Explanation Builder. The only exception is observed on explaining predictions on DB50K made with CONVE. Moreover, IMAGINE, when equipped with simulation or bisimulation, always performs on par or better than its version with no summarization in the Explanation Builder.

In Tab. 4 we report the effectiveness metrics for the experiments measuring the worsening of LP performance. In this setting, KELPIE and KELPIE++ consistently achieve a larger worsening than W-IMAGINE. We remark that the explanations computed by IMAGINE can be complementary to those computed as SCEs. Indeed, IMAGINE outperforms the methods providing SCEs on non top-ranked test triples. In contrast, W-IMAGINE as an adaption of IMAGINE to be

[3] `https://github.com/rbarile17/imagine`

Table 3: Outcomes of the experiments: *improvement* setting

| KGE model | Method | Summarization | DB100K | | DB50K | | YAGO4-20 | |
|---|---|---|---|---|---|---|---|---|
| | | | *ΔH@1* | *ΔMRR* | *ΔH@1* | *ΔMRR* | *ΔH@1* | *ΔMRR* |
| ComplEx | Imagine | – | 0.510 | 0.458 | **0.580** | **0.536** | 0.520 | 0.501 |
| | Imagine | *bisimulation* | 0.510 | 0.458 | **0.580** | **0.536** | 0.520 | 0.501 |
| | Imagine | *simulation* | **0.540** | **0.490** | 0.450 | 0.433 | **0.560** | **0.537** |
| | I-Kelpie | – | 0.130 | 0.100 | 0.030 | -0.029 | 0.080 | 0.037 |
| | I-Kelpie++ | *bisimulation* | 0.130 | 0.100 | 0.030 | -0.029 | 0.080 | 0.037 |
| | I-Kelpie++ | *simulation* | 0.090 | 0.062 | 0.030 | -0.037 | 0.100 | 0.061 |
| ConvE | Imagine | – | 0.420 | 0.347 | 0.040 | 0.012 | 0.280 | 0.285 |
| | Imagine | *bisimulation* | 0.420 | 0.347 | 0.040 | 0.012 | 0.280 | 0.285 |
| | Imagine | *simulation* | **0.430** | **0.378** | 0.030 | 0.008 | **0.510** | **0.502** |
| | I-Kelpie | – | 0.100 | 0.036 | 0.040 | 0.014 | 0.050 | 0.036 |
| | I-Kelpie++ | *bisimulation* | 0.100 | 0.036 | 0.040 | 0.014 | 0.050 | 0.036 |
| | I-Kelpie++ | *simulation* | 0.140 | 0.081 | **0.050** | **0.025** | 0.040 | 0.030 |
| TransE | Imagine | – | **0.140** | 0.165 | 0.360 | 0.292 | 0.330 | 0.332 |
| | Imagine | *bisimulation* | **0.140** | 0.165 | 0.360 | 0.292 | 0.330 | 0.332 |
| | Imagine | *simulation* | **0.140** | **0.184** | **0.410** | **0.333** | **0.400** | **0.409** |
| | I-Kelpie | – | 0.080 | 0.073 | 0.090 | 0.036 | 0.030 | 0.002 |
| | I-Kelpie++ | *bisimulation* | 0.080 | 0.073 | 0.090 | 0.036 | 0.030 | 0.002 |
| | I-Kelpie++ | *simulation* | 0.040 | 0.023 | 0.100 | 0.039 | 0.020 | 0.001 |

employed for generating ACEs that impact top-ranked test triples underperforms the methods providing SCEs. The latter observation highlights that removing triples has a greater impact on correct predictions than *additional triples*, but also indicates that *additional triples* do not introduce such significant noise that adding them degrades performance more than removing existing triples. Finally, Imagine, takes on average 79 seconds to explain a prediction on the environment that we employed.

Moving to a qualitative analysis, we now illustrate typical examples of explanation output, for both non top-ranked triples and those that are ranked first. We report the explanations produced by Imagine and I-Kelpie for a non top-ranked test triple. Specifically, we employ the triple $\langle$*John Deacon*, *member of*, *Queen (band)*$\rangle$ found in the experiments with TransE on YAGO4-20.

Imagine explains the prediction by returning the *additional triples* identified by the dashed edges in Fig. 2a. Such an explanation underscores that the LP method mostly alters the rank of the triple about band membership, if fed with information on the participation in albums and the formation of the band. In contrast, I-Kelpie returns the triples identified by edges with strike-through labels in Fig. 2b. Hence, the LP method change its outcomes mostly based on the predicates: *occupation*, *given name*, and *alumni of*. However, Imagine seems able to hypothesize connections with the object of the prediction for entities lacking such information.

Table 4: Outcomes of the experiments: *worsening* setting

| KGE model | Method | Summarization | DB100K | | DB50K | | YAGO4-20 | |
|---|---|---|---|---|---|---|---|---|
| | | | *ΔH@1* | *ΔMRR* | *ΔH@1* | *ΔMRR* | *ΔH@1* | *ΔMRR* |
| ComplEx | W-Imagine | – | -0.280 | -0.203 | -0.210 | -0.156 | -0.390 | -0.268 |
| | W-Imagine | *bisimulation* | -0.280 | -0.203 | -0.210 | -0.156 | -0.390 | -0.268 |
| | W-Imagine | *simulation* | -0.310 | -0.247 | -0.280 | -0.224 | -0.450 | -0.340 |
| | Kelpie | – | -0.710 | -0.614 | -0.980 | **-0.960** | -0.610 | -0.469 |
| | Kelpie++ | *bisimulation* | -0.710 | -0.614 | -0.980 | **-0.960** | -0.610 | -0.469 |
| | Kelpie++ | *simulation* | **-0.780** | **-0.673** | **-0.990** | -0.952 | **-0.650** | **-0.540** |
| ConvE | W-Imagine | – | -0.180 | -0.112 | -0.060 | -0.040 | -0.240 | -0.170 |
| | W-Imagine | *bisimulation* | -0.180 | -0.112 | -0.060 | -0.040 | -0.240 | -0.170 |
| | W-Imagine | *simulation* | -0.170 | -0.112 | -0.040 | -0.022 | -0.290 | -0.208 |
| | Kelpie | – | -0.230 | -0.185 | -0.090 | -0.057 | -0.310 | -0.248 |
| | Kelpie++ | *bisimulation* | -0.230 | -0.185 | -0.090 | -0.057 | -0.310 | -0.248 |
| | Kelpie++ | *simulation* | **-0.310** | **-0.246** | **-0.190** | **-0.121** | **-0.400** | **-0.329** |
| TransE | W-Imagine | – | -0.560 | -0.394 | -0.290 | -0.196 | -0.480 | -0.393 |
| | W-Imagine | *bisimulation* | -0.560 | -0.394 | -0.290 | -0.196 | -0.480 | -0.393 |
| | W-Imagine | *simulation* | -0.500 | -0.357 | -0.470 | -0.361 | -0.400 | -0.312 |
| | Kelpie | – | **-0.710** | **-0.587** | **-0.760** | **-0.666** | -0.610 | -0.501 |
| | Kelpie++ | *bisimulation* | **-0.710** | **-0.587** | **-0.760** | **-0.666** | -0.610 | -0.501 |
| | Kelpie++ | *simulation* | -0.680 | **-0.581** | -0.710 | -0.606 | **-0.680** | **-0.580** |

Conversely, we report the explanations by W-Imagine and Kelpie for a given test set triple ranked first by the LP method. In particular, we consider the prediction $\langle$*Renato Scarpa*, *has occupation*, *actor*$\rangle$. Kelpie explains it by returning the triple $\langle$*The Talented Mr. Ripley (film)*, *actor*, *Renato Scarpa*$\rangle$. This suggests that the LP method predicted *actor* as the occupation for *Renato Scarpa* because, during training, it had seen a triple stating his occupation as *actor* in a film. Instead, W-Imagine generates the *additional triples* identified by dashed edges in Fig. 3. This shows that the LP method switches its prediction to another occupation when provided with triples stating different occupations in different works. Thus, W-Imagine seems to be able to complement the insight given by Kelpie as it provides a further example of the dependence of occupation on triples stating such an occupation.

## 6 Conclusions

We introduced Imagine, a novel complementary approach to provide CEs for predictions made on KGs. It is based on the generation of plausible and meaningful additional triples. It generates ACEs consisting of sets of *additional triples* having the greatest impact on the predictions. We performed an experimental evaluation on three datasets and three representative LP models for demonstrating that the generated ACEs are valuable, i.e., able to improve the LP performance. We also qualitatively assessed the differences between the alternative CEs in each setting.

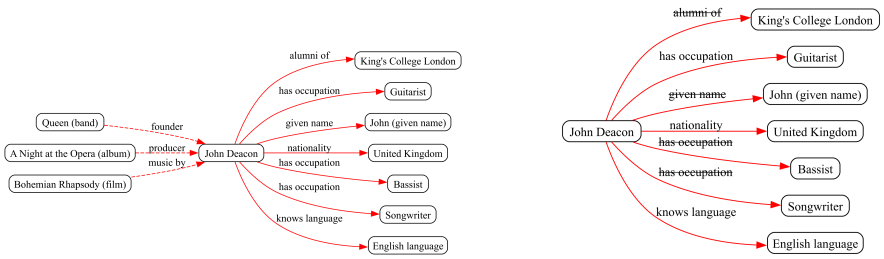



Fig. 2: Subgraphs with an ACE (a) and an SCE (b) for the prediction $\langle \textit{John Deacon}, \textit{member of}, \textit{Queen (band)} \rangle$. ACE triples are identified by dashed edges, SCE triples by edges with strike-through labels.

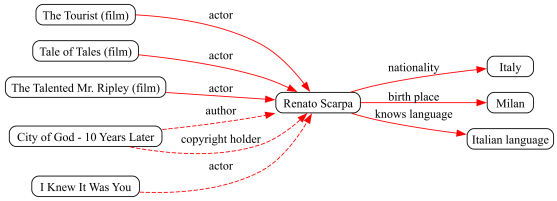



Fig. 3: Subgraph featuring *Renato Scarpa* for the prediction $\langle \textit{Renato Scarpa}, \textit{has occupation}, \textit{actor} \rangle$. ACE triples are indicated by dashed edges.

A natural extension of this work would be a unified approach that generates explanations that include both existing and additional triples, thus integrating SCEs and ACEs. In addition, we aim to enhance the Triple Builder component used to generate *additional triples*. For example, we may use a probabilistic generative model, such as the one proposed in [24], which would allow sampling triples from the encoded distribution. In addition, we plan to investigate the impact of distinguishing between *additional triples* that cannot be proven neither true nor false and those that can be proven false (negative statements) according to the KG schema. Finally, we may involve users in the qualitative analysis.

**Acknowledgments.** This work was partially supported by project *FAIR - Future AI Research* (PE00000013), spoke 6 - Symbiotic AI (`https://future-ai-research.it/`) under the PNRR MUR program funded by the European Union - NextGenerationEU, by PRIN project *HypeKG - Hybrid Prediction and Explanation with Knowledge Graphs* (Prot. 2022Y34XNM, CUP H53D23003700006) under the PNRR MUR program funded by the European Union - NextGenerationEU, and by project *DIXTI - DIgging into eXplainable soluTIons for link prediction on Knowledge Graphs using large language models* under ISCRA program funded by CINECA.

**Disclosure of Interests.** The authors declare to have no competing interests.

## References


1. Amador-Domínguez, E., Serrano, E., Manrique, D.: GEnI: A framework for the generation of explanations and insights of knowledge graph embedding predictions. Neurocomputing **521**, 199–212 (2023). https://doi.org/10.1016/j.neucom.2022.12.010
2. Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., Ives, Z.: DBpedia: A Nucleus for a Web of Open Data. In: Hutchison, D., Kanade, T., Kittler, J., Kleinberg, J.M., Mattern, F., Mitchell, J.C., Naor, M., Nierstrasz, O., Pandu Rangan, C., Steffen, B., Sudan, M., Terzopoulos, D., Tygar, D., Vardi, M.Y., Weikum, G., Aberer, K., Choi, K.S., Noy, N., Allemang, D., Lee, K.I., Nixon, L., Golbeck, J., Mika, P., Maynard, D., Mizoguchi, R., Schreiber, G., Cudré-Mauroux, P. (eds.) The Semantic Web, vol. 4825, pp. 722–735. Springer Berlin Heidelberg, Berlin, Heidelberg (2007). https://doi.org/10.1007/978-3-540-76298-0_52
3. Baltatzis, V., Costabello, L.: KGEx: Explaining Knowledge Graph Embeddings via Subgraph Sampling and Knowledge Distillation (Oct 2023)
4. Barile, R., d'Amato, C., Fanizzi, N.: Explanation of Link Predictions on Knowledge Graphs via Levelwise Filtering and Graph Summarization. In: Meroño Peñuela, A., Dimou, A., Troncy, R., Hartig, O., Acosta, M., Alam, M., Paulheim, H., Lisena, P. (eds.) Proceedings of the 26th European Semantic Web Conference (ESWC 2024), vol. 14664, pp. 180–198. Springer Nature Switzerland, Cham (2024). https://doi.org/10.1007/978-3-031-60626-7_10
5. Betz, P., Meilicke, C., Stuckenschmidt, H.: Adversarial Explanations for Knowledge Graph Embeddings. In: IJCAI. vol. 2022, pp. 2820–2826 (2022)
6. Bhowmik, R., De Melo, G.: Explainable Link Prediction for Emerging Entities in Knowledge Graphs. In: Pan, J.Z., Tamma, V., d'Amato, C., Janowicz, K., Fu, B., Polleres, A., Seneviratne, O., Kagal, L. (eds.) The Semantic Web – ISWC 2020, vol. 12506, pp. 39–55. Springer International Publishing, Cham (2020). https://doi.org/10.1007/978-3-030-62419-4_3
7. Bollacker, K., Cook, R., Tufts, P.: Freebase: A shared database of structured general human knowledge. In: AAAI. vol. 7, pp. 1962–1963 (2007)
8. Bordes, A., Usunier, N., Garcia-Duran, A., Weston, J., Yakhnenko, O.: Translating embeddings for modeling multi-relational data. Advances in neural information processing systems **26** (2013)
9. Byrne, R.M.: Counterfactuals in Explainable Artificial Intelligence (XAI): Evidence from Human Reasoning. In: IJCAI. pp. 6276–6282 (2019)
10. Čebirić, Š., Goasdoué, F., Kondylakis, H., Kotzinos, D., Manolescu, I., Troullinou, G., Zneika, M.: Summarizing semantic graphs: A survey. The VLDB journal **28**, 295–327 (2019)
11. d'Amato, C., Masella, P., Fanizzi, N.: An Approach Based on Semantic Similarity to Explaining Link Predictions on Knowledge Graphs. In: IEEE/WIC/ACM International Conference on Web Intelligence. pp. 170–177. ACM, ESSENDON VIC Australia (Dec 2021). https://doi.org/10/gtmzqz
12. Dettmers, T., Minervini, P., Stenetorp, P., Riedel, S.: Convolutional 2d knowledge graph embeddings. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 32 (2018). https://doi.org/10/gqjp9q
13. Ding, B., Wang, Q., Wang, B., Guo, L.: Improving Knowledge Graph Embedding Using Simple Constraints. In: Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). pp. 110–121 (2018)

14. Dong, X.L.: Building a broad knowledge graph for products. In: 2019 IEEE 35th International Conference on Data Engineering (ICDE). pp. 25–25. IEEE (2019). https://doi.org/10.1109/ICDE.2019.00010
15. Glimm, B., Horrocks, I., Motik, B., Stoilos, G., Wang, Z.: HermiT: An OWL 2 Reasoner. Journal of Automated Reasoning **53**(3), 245–269 (Oct 2014). https://doi.org/10.1007/s10817-014-9305-1
16. Guidotti, R.: Counterfactual explanations and how to find them: Literature review and benchmarking. Data Mining and Knowledge Discovery (Apr 2022). https://doi.org/10.1007/s10618-022-00831-6
17. Guidotti, R., Monreale, A., Ruggieri, S., Turini, F., Giannotti, F., Pedreschi, D.: A Survey of Methods for Explaining Black Box Models. ACM Computing Surveys **51**(5), 1–42 (Sep 2019). https://doi.org/10/gfgt9s
18. Hogan, A., Gutierrez, C., Cochez, M., de Melo, G., Kirrane, S., Polleres, A., Navigli, R., Ngonga Ngomo, A.C., Rashid, S.M., Schmelzeisen, L., Staab, S., Blomqvist, E., d'Amato, C., Labra Gayo, J.E., Neumaier, S., Rula, A., Sequeda, J., Zimmerman, A.: Knowledge Graphs. No. 22 in Synthesis Lectures on Data, Semantics, and Knowledge, Springer (Jun 2022). https://doi.org/10.2200/S01125ED1V01Y202109DSK022, https://kgbook.org/
19. Ismaeil, Y., Stepanova, D., Tran, T.K., Blockeel, H.: FeaBI: A Feature Selection-Based Framework for Interpreting KG Embeddings. In: Payne, T.R., Presutti, V., Qi, G., Poveda-Villalón, M., Stoilos, G., Hollink, L., Kaoudi, Z., Cheng, G., Li, J. (eds.) The Semantic Web – ISWC 2023, vol. 14265, pp. 599–617. Springer Nature Switzerland, Cham (2023). https://doi.org/10.1007/978-3-031-47240-4_32
20. Koh, P.W., Liang, P.: Understanding black-box predictions via influence functions. In: International Conference on Machine Learning. pp. 1885–1894. PMLR (2017)
21. Krishnan, N.A., Rivero, C.R.: A Model-Agnostic Method to Interpret Link Prediction Evaluation of Knowledge Graph Embeddings. In: Proceedings of the 32nd ACM International Conference on Information and Knowledge Management. pp. 1107–1116. ACM, Birmingham United Kingdom (Oct 2023). https://doi.org/10.1145/3583780.3614763
22. Kumar-Singh, N., Polleti, G., Paliwal, S., Hodos-Nkhereanye, R.: LinkLogic: A New Method and Benchmark for Explainable Knowledge Graph Predictions (Jun 2024)
23. Law, J.: Robust statistics—the approach based on influence functions (1986)
24. Loconte, L., Di Mauro, N., Peharz, R., Vergari, A.: How to Turn Your Knowledge Graph Embeddings into Generative Models. Advances in Neural Information Processing Systems **36** (2024)
25. Lundberg, S.M., Lee, S.I.: A unified approach to interpreting model predictions. Advances in neural information processing systems **30** (2017)
26. Ma, T., Song, X., Tao, W., Li, M., Zhang, J., Pan, X., Lin, J., Song, B., Zeng, x.: KGExplainer: Towards Exploring Connected Subgraph Explanations for Knowledge Graph Completion (Apr 2024)
27. Nováček, V., Mohamed, S.K.: Predicting polypharmacy side-effects using knowledge graph embeddings. AMIA Summits on Translational Science Proceedings **2020**, 449 (2020)
28. Pellissier Tanon, T., Weikum, G., Suchanek, F.: YAGO 4: A Reason-able Knowledge Base. In: Harth, A., Kirrane, S., Ngonga Ngomo, A.C., Paulheim, H., Rula, A., Gentile, A.L., Haase, P., Cochez, M. (eds.) The Semantic Web, vol. 12123, pp. 583–596. Springer International Publishing, Cham (2020). https://doi.org/10.1007/978-3-030-49461-2_34

29. Pezeshkpour, P., Irvine, C.A., Tian, Y., Singh, S.: Investigating Robustness and Interpretability of Link Prediction via Adversarial Modifications. In: Proceedings of NAACL-HLT. pp. 3336–3347 (2019)
30. Rossi, A., Barbosa, D., Firmani, D., Matinata, A., Merialdo, P.: Knowledge Graph Embedding for Link Prediction: A Comparative Analysis. ACM Transactions on Knowledge Discovery from Data **15**(2), 1–49 (Apr 2021). https://doi.org/10/gjhzcz
31. Rossi, A., Firmani, D., Merialdo, P., Teofili, T.: Explaining Link Prediction Systems based on Knowledge Graph Embeddings. In: Proceedings of the 2022 International Conference on Management of Data. pp. 2062–2075. ACM, Philadelphia PA USA (Jun 2022). https://doi.org/10/gqjfg6
32. Shi, B., Weninger, T.: Open-world knowledge graph completion. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 32 (2018)
33. Singhal, A.: Introducing the Knowledge Graph: Things, not strings. https://blog.google/products/search/introducing-knowledge-graph-things-not/ (May 2012)
34. Trouillon, T., Welbl, J., Riedel, S., Gaussier, É., Bouchard, G.: Complex embeddings for simple link prediction. In: International Conference on Machine Learning. pp. 2071–2080. PMLR (2016)
35. Ying, Z., Bourgeois, D., You, J., Zitnik, M., Leskovec, J.: Gnnexplainer: Generating explanations for graph neural networks. Advances in neural information processing systems **32** (2019)
36. Zhang, H., Zheng, T., Gao, J., Miao, C., Su, L., Li, Y., Ren, K.: Data poisoning attack against knowledge graph embedding. In: Proceedings of the 28th International Joint Conference on Artificial Intelligence. pp. 4853–4859 (2019)
37. Zhang, W., Deng, S., Wang, H., Chen, Q., Zhang, W., Chen, H.: XTransE: Explainable Knowledge Graph Embedding for Link Prediction with Lifestyles in e-Commerce. In: Wang, X., Lisi, F.A., Xiao, G., Botoeva, E. (eds.) Semantic Technology, vol. 1157, pp. 78–87. Springer Singapore, Singapore (2020). https://doi.org/10.1007/978-981-15-3412-6_8
38. Zhang, W., Paudel, B., Zhang, W., Bernstein, A., Chen, H.: Interaction Embeddings for Prediction and Explanation in Knowledge Graphs. In: Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining. pp. 96–104. ACM, Melbourne VIC Australia (Jan 2019). https://doi.org/10/ggzfzp
39. Zhao, D., Wan, G., Zhan, Y., Wang, Z., Ding, L., Zheng, Z., Du, B.: KE-X: Towards subgraph explanations of knowledge graph embedding based on knowledge information gain. Knowledge-Based Systems **278**, 110772 (2023)